Operator/System Communication : An Optimizing Decision Tool

Tarek M. Sobh and Tarek Alameldin

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania, Philadelphia, PA 19104

Tarek M. Sobh and Tarek K. Alameldin "Operator/system communication: an optimizing decision tool", Proc. SPIE 1388, Mobile Robots V, 524 (March 1, 1991); doi:10.1117/12.25495;

Operator/System Communication : An Optimizing Decision Tool

Tarek M. Sobh and Tarek Alameldin

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania, Philadelphia, PA 19104

Abstract

In this paper we address the problem of operator/system communication. In particular, we discuss the issue of efficient and adaptive transmission mechanisms over possible physical links. We develop a tool for making decisions regarding the flow of control sequences and data from and to the operator. The issue of compression is discussed in details, a decision box and an optimizing tool for finding the appropriate thresholds for a decision are developed. Physical parameters like the data rate, bandwidth of the communication medium, distance between the operator and the system, baud rate, levels of discretization, signal to noise ratio and propagation speed of the signal are taken into consideration while developing our decision system. Theoretical analysis is performed to develop mathematical models for the optimization algorithm. Simulation models are also developed for testing both the optimization and the decision tool box.

1 Introduction

Data which is transmitted over a communication medium between the operator and the system, which might be a mobile robot, contains some form of redundancy. This redundancy can be exploited to make economical use of the storage media or to reduce the cost of transferring the data and commands over the communication network. One of the basic issues in the design of the presentation layer is to decide whether data is to be compressed before transmission or not. Many factors may affect making this decision, one of the most important ones is the cost factor. It should be clear that the more information (bits) that are sent over the network, the more money that one has to pay. Compressing data before sending it will often help reduce the cost.

Some other factors may affect the decision of compression. The time factor may be the influencing one, in fact, one should not forget that there is the overhead of the compression and decompression algorithms at the sender and at the receiver hosts. This overhead is both in time and money, as the CPU is used for running the algorithms. Thus, the designer always faces the problem of when should one compress the data. The parameters which may affect this decision might be the parameters of the physical communication medium, they might also be the parameters of the compression algorithm used or both.

The decision that the design engineer will have to make might be a decision to compress or not given a certain fixed compression and physical medium parameters, or it might be a decision to compress depending on the value of one or more of the parameters (i.e., to compress if a certain threshold is met). In our work, we try to develop a tool for making such a decision, we choose the time to be the criteria for making compression decision, where the time is both for the compression overhead and for transmission. We develop a yes/no decision box given some fixed parameters and an optimizing decision box for finding the threshold for one varying parameter while fixing the others. Theoretical analysis is performed and also simulations for different sets of data and a decision (or a threshold) is output for each kind of analysis.

2 Physical and Compression Parameters

The parameters of the communication medium between two hosts will affect the decision regarding compression. Whether the medium be a coaxial cable, a fiber optic cable or air (in the case of radio transmissions) it can always be completely specified by parameterizing it. The parameters that may help in determining the transmission time can be listed as follows :

- The data rate in bits per second. { D bps }
- The band width of the medium in hertz. { B Hz }
- The distance between the two hosts under consideration in meters. { L m }
- The levels of discretization. { l }
- The baud rate in changes per second. { b }
- The signal to noise ratio in decibels. { S dB }
- The propagation speed of the signal in the medium, in meters per second. { P m/s }

It should be noticed that there is redundancy in expressing the time for transmission using all those parameters and the number of bits sent. For example, it is sufficient to use the number of bits and the data rate to express the time. However, if the data rate is not available we can use the baud rate, the levels of discretization and the data size, or alternatively we can use Shanon's maximum data rate bound, thus using the band width, the signal to noise ratio and the data size to find an expression for the minimum time for transmission.

The other set of parameters that is involved with the computation of the time for transmitting a certain amount of data is the set of the compression algorithm parameters. The CPU run time as a function of the size of data input to the algorithm is one of those parameters. The expected compression ratio, which actually depends on what *type* of data to be transmitted is the second compression parameter of concern.

3 Mathematical Formulation

The problem can be mathematically formulated by trying to find the cost of sending a certain number of bits from a host to another. The cost will be assumed to be the time through which the channel will be kept busy sending the data plus the time that will take the CPU to perform the compression and decompression on the data that are required to be transmitted. One can use a weight in the cost expression to denote that, for example, the cost for utilizing the network cable for one second is X times the cost for utilizing the CPU for one second. Thus, the expression for the cost function may be written as :

$$\text{Transmission time} + X \times \text{CPU computation time}$$

where X is the ratio between the cost of using the network for one unit time and the cost of one unit CPU time.

3.1 The Transmission Time

If we make the assumption that we only have two hosts connected directly and ignore the other overheads of the protocol to be used, the time needed to transmit N bits from a host to another can be written as a mathematical expression in terms of the physical medium parameters. For our implementation we are going to develop the transmission expression in four different ways, using four different sets of physical parameters, where each set individually is sufficient to specify the transmission time $T1$ completely.

3.1.1 Formulation Using the Data rate

The time required for transmitting N bits can be formulated as follows :

$$T1 = \frac{N}{D} + \frac{L}{P}$$

where D is the data rate in bits per second, L is the distance between the two hosts and P is the signal propagation speed. The first term is for the emission of N bits from the sender and the second term is the time for the last bit to reach the receiver.

3.1.2 Formulation Using the baud rate

The time required for transmitting N bits can be formulated as follows :

$$T1 = \frac{N}{b \log_2 l} + \frac{L}{P}$$

where b is the baud rate in changes per second, l is the number of levels of discretization, L is the distance between the two hosts and P is the signal propagation speed. The first term is for the emission of N bits from the sender and the second term is the time for the last bit to reach the receiver.

3.1.3 Formulation Using the band width

The time required for transmitting N bits can be formulated as follows :

$$T1 = \frac{N}{2B \log_2 l} + \frac{L}{P}$$

where B is the band width in Hertz, l is the number of levels of discretization, L is the distance between the two hosts and P is the signal propagation speed. The first term is for the emission of N bits from the sender and the second term is the time for the last bit to reach the receiver. In this case, there is assumed to be no noise whatsoever, we are assuming the maximum possible data rate.

3.1.4 Formulation Using the Signal to Noise Ratio

The time required for transmitting N bits can be formulated as follows :

$$T1 = \frac{N}{B \log_2(1 + 10^{\frac{S}{10}})} + \frac{L}{P}$$

where B is the band width in Hertz, S is the signal to noise ratio in decibels, L is the distance between the two hosts and P is the signal propagation speed. The first term is for the emission of N bits from the sender and the second term is the time for the last bit to reach the receiver. In this case Shanon's maximum data rate of a noisy channel is assumed.

3.2 The Compression and Decompression Times

The run times of the algorithm for compression and decompression can be expressed as a function of the size of the input in terms of machine cycles. That is, the compression time can be expressed as **T2**(M) where M is the size of data that is input to the compression algorithm.

3.3 Total Cost Without Using Compression

The total time to send N bits without using compression would then simply be equal to the transmission time, thus it equals one of the four expressions discussed previously. The total cost is considered to be only the time during which the communication channel is utilized.

3.4 Total Cost Using Compression

The total cost for transmitting a sequence of bits using compression will be assumed to be a weighted combination of the times for transmission and the times for compression and decompression. Thus, if we assume the compression ratio of the algorithm to be equal to **R**, and X is the ratio between the cost of using the network for one unit time and the cost of one unit CPU time and if we also assume a variable page size, i,e. compression is to be performed before each transmission of a block of size M of data, the total cost to be incurred (when we express the transmission time in terms of the data rate) can be written as :

$$C = \frac{\frac{1}{\mathbf{R}}M}{D} + \frac{L}{P} + X(\mathbf{f1}(M) + \mathbf{f2}(\frac{1}{\mathbf{R}}M))$$

where **f1** and **f2** are the compression and decompression runtime functions (in terms of the input size).
Similarly, the total cost can be written for the other physical medium sets of parameters as :

$$C = \frac{\frac{1}{\mathbf{R}}M}{b\log_2 l} + \frac{L}{P} + X(\mathbf{f1}(M) + \mathbf{f2}(\frac{1}{\mathbf{R}}M))$$

or

$$C = \frac{\frac{1}{\mathbf{R}}M}{2B\log_2 l} + \frac{L}{P} + X(\mathbf{f1}(M) + \mathbf{f2}(\frac{1}{\mathbf{R}}M))$$

or

$$C = \frac{\frac{1}{\mathbf{R}}M}{B\log_2(1+10^{\frac{S}{10}})} + \frac{L}{P} + X(\mathbf{f1}(M) + \mathbf{f2}(\frac{1}{\mathbf{R}}M))$$

4 Compression Algorithms

The methods to compress data have been studied for many years. However, several problems have prevented the widespread integration of compression methods into computer systems for automatic data compression. These problems include poor runtime execution preventing high data rates, lack of flexibility in the compression procedures to deal with most types of redundancies and storage management problems in dealing with storage of blocks of data of unpredictable length. In most cases a method presents some subset of these problems and therefore is restricted to applications

where its inherent weaknesses do not result in critical inefficiencies. In this section we shall review the different forms of redundancies that can be taken advantage of for compression and then look at some of the approaches taken towards this end. Then we shall present a method due to Welch [3] which avoids many of the drawbacks of most of the methods.

4.1 Kinds of Redundancies

There are four major types of redundancies that we shall mention here. The forms of redundancies discussed are not independent of each other but overlap to some extent. There are some other forms of redundancies too, but the ones we are going to discuss are the major ones.

In different types of data, some of the characters are used more frequently than others. For example, in English text we see space and 'e' more frequent than any other character and special characters are used a lot less frequently. Generally speaking, all of the string combinations might not be used in a specific data set, resulting in the possibility of reducing the number of bits per combination. This kind of redundancy is due to character distribution.

The repetition of string patterns is another form of redundancy found in some of the cases. For example, the sequence of blank spaces is very common in some kind of data files. In such cases the message can usually be encoded more compactly rather than by repeating the string pattern.

In a certain type of data set, certain sequences of characters might appear very frequently. Some pairs may be used with higher frequency than the individual probabilities of the letters in these pairs would imply. Therefore, these pairs could be encoded using fewer bits than the combined combinations of the two characters formed by joining together the individual combinations for the two characters. Likewise, the unusual pairs, can be encoded using very long bit patterns to achieve better utilization.In some data sets, certain strings or numbers consistently appear at a predictable position. This is called Positional redundancy. It is a form of partial redundancy that can be exploited in encoding.

4.2 Methods of Compression

Using the discussion on redundancy types as our basis, we shall discuss several practical compression methods, and then choose one of them and use it for our simulation.

4.2.1 Huffman Encoding

This is the most popular compression method. It translates the fixed-size pieces of input data into variable-length symbols. The standard Huffman encoding procedure prescribes a way to assign codes to input symbols such that each code length in bits is approximately $\log_2(SymbolProbability)$. Where symbol probability is the relative frequency of occurrence of a given symbol (expressed as a probability). Huffman encoding has certain problems. The first problem is that the size of input symbols is restricted by the size of the translation table. If a symbol is one eight-bit byte, then a table of 256 entries is sufficient. However, such a table limits the degree of compression that can be achieved. If the size of the input symbols is increased to two bytes each, the compression degree would be increased. However, such encoding would require a table of 64K entries which may be a high cost.

The second problem with Huffman encoding is the complexity of the decompression process. The translation table is essentially a binary tree, in that, the interpretation of each code proceeds bit by

bit and a translation subtable is chosen depending on whether the bit is zero or one. This basically means a logic decision on every bit which can create a system bottle neck.

The third problem with Huffman encoding is the need to know the frequency distribution of characters in the input data which is not well known for all kinds of data. A common solution is to do two passes on the data, one to find the frequency distribution ordering and the other is to to do the encoding. This process may be done block wise to adapt to the changes in data. This is not very efficient although it might be acceptable.

4.2.2 Run-length encoding

Repeated sequences of identical characters can be encoded as a count field along with the repeated character. However, the count fields have to be distinguished from the normal character fields which might have the same bit pattern as the count fields. A possible solution is to use a special character to mark the count field. This might be suitable for ASCII text, but not when there are arbitrary bit patterns such as in the case of binary integers. Typically, three characters are required to mark a sequence of an identical character. So, this will not be useful for sequences of length three or less.

4.2.3 Programmed Compression

In formatted data files, several techniques are used to do compression. Unused blank or zero spaces are eliminated by making fields variable in length and using an index structure with pointers to each field position. Predicted fields are compactly encoded by a code table. Programmed compression cannot effectively handle character distribution redundancy and is therefore a nice complement of Huffman encoding. The programmed compression has several drawbacks. First it is usually done by the application programmers adding to the software development cost. The type of decompression used requires a knowledge of the record structure and the code tables. The choice of field sizes and code tables may complicate or inhibit later changes to the data structure making the software more expensive to maintain.

4.2.4 Adaptive Compression

The Lempel-Ziv [4,5] and related methods fall into this category. Fixed length codes are used for variable-length strings such that the probability of occurrence of all selected strings is almost equal. This implies that the strings comprising of more frequently occurring symbols will contain more symbols than those strings having more of the infrequent symbols. This type of algorithm exploits character frequency redundancy, character repetitions, and high-usage pattern redundancy although it is usually not effective on positional redundancy. The algorithm is adaptive in the sense that it starts with an empty translation table and builds the table as the compression proceeds. This type of algorithm is a one pass procedure and usually requires no prior information of the type of data. Such algorithm, gives poor compression results in the initial part of the data set; as a result the data set should be long enough for the procedure to establish enough symbol frequency experience to achieve a good compression degree over the whole data set. On the other hand, most finite implementations of an adaptive algorithm loose the ability to adapt after certain length of the input which means that if the input's redundancy characteristics vary over its length, the compression degree declines if the input length significantly exceeds the adaptive range of the compression implementation.

We have chosen a variation of the Lempel-Ziv procedure which is called LZW due to Welch [3]. This method retains the adaptive characteristics of the Lempel-Ziv procedure but is distinguished by its

very simple logic which yields relatively inexpensive implementations and a potential of very fast execution.

4.3 The LZW Algorithm

The LZW algorithm is organized around a translation table, referred to here as a string table, that maps strings of input characters into fixed length codes. The use of 12-bit codes is common. The LZW table has the prefix property that if wK is a string in the table and w is a string and K is a character, then w is also in the table. K is called the extension character on the prefix string w. The string table may be initialized to contain all single-character strings.

The LZW table, at any time in the compression process, contains strings that have been encountered previously in the message being compressed. In other words, it contains the running sample of strings in the message, so the available strings reflect the statistics of the message. The strings added to the table are determined by this parsing: each parsed input string extended by its next input character forms a new string added to the string table. Each such string is assigned a unique identifier, namely its code value. In precise terms, this is the algorithm :

Initialize table to contain single-character strings.
Read first input character $\Longrightarrow$ prefix string w.
Step: Read next input character K
If no such K (input exhausted): code(w)$\Longrightarrow$ output ;EXIT
If wK exists in string table: $wK \Longrightarrow w$; repeat step.
else wK not in string table: code(w)$\Longrightarrow$ output;
$wK \Longrightarrow$ string table;
$K \Longrightarrow w$; repeat step.

The algorithm is quite simple and can have a very fast implementation. The main concern in the implementation is storing the string table which can be very large. However, it can be made tractable by representing each string by its prefix identifier and extension character. This will give a table of fixed length entries.

The basic algorithm for decompression is as follows :

Decompression: Read first input code$\Longrightarrow$ CODE $\Longrightarrow$ OLDcode;
with CODE = code(K), $K \Longrightarrow$ output;
Next Code: Read next input code $\Longrightarrow$ CODE $\Longrightarrow$ INcode;
If no new code : EXIT, else:
Next Symbol: If CODE = code(wK): $K \Longrightarrow$ output;
code(w) $\Longrightarrow$ CODE;
Repeat Next Symbol
else if CODE = code(K)$K \Longrightarrow$ output;
OLDcode, $K \Longrightarrow$ string table;
INcode $\Longrightarrow$ OLDcode;
Repeat Next Code.

The decompressor logically uses the same string table as the compressor and similarly constructs it as the message is translated. Each received code value is translated by way of the string table into

a prefix string and extension character. The extension character is pulled off and the prefix string is decomposed into its prefix and extension. This operation is recursive until the prefix string is a single character, which completes decompression of that code. Each update to the string table is made for each code received (except the first one). When a code has been translated , its final character is used as the extension character, combined with the prior string, to add a new string to the string table. This new string is assigned a unique code value, which is the same code that the compression assigned to that string. In this way, the decompressor incrementally reconstructs the same string table that the compressor used.

5 Comparing the Models

The goal of our mathematical formulations and modeling is to perform one of two basic tasks, the first one is to decide whether to use compression or not given a certain set of fixed parameters { for compression, decompression and the physical medium } , the other is to decide the threshold for a specific varying parameter before which we should perform compression and after which we should not perform compression.

Two independent situations can arise in our formulation, in the first one, we can consider the protocol in which the communication to take place is a one of varying page length. In this case, the compression is performed for one "chunk" of data at a time and is immediately sent after that. In the other case, the protocol may have a fixed page size and thus the compression is performed for large files and the compressed data is sent one page at a time. Thus comparing the two models for decision making and optimizing parameters can be performed for each one of these situations separately. It should also be noticed that there might exist hypothetical bounds and average values for the run times and compression ratios for the compression and decompression algorithms.

5.1 Using a Varying-Length Page

The problem in this case is to either make a decision regarding compression or to optimize a parameter, the four different representations for the transmission time can each be used to formulate and express the total cost incurred in the compression and uncompression modes. In the decision problem, we choose the scheme to have the less cost. In the optimization problems we find the range for a certain parameter such that, for example, compression is a better technique, by solving the inequality.

Assuming that we use the LZW algorithm, characters are 8 bits each, the machine's cycle rate is w cycles per second, the data size to be compressed is M bits and the compression ratio is $\mathbf{R}$. The algorithm runtime can be formulated as :

$$\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}$$

The following inequality can be formed for the model using the data rate as the physical parameter, for cost of the compressed mode to be less than the cost of the uncompressed mode .

$$\frac{\frac{1}{\mathbf{R}}M}{D} + \frac{L}{P} + X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}} + \frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right) \leq \frac{M}{D} + \frac{L}{P}$$

For the model using the baud rate

$$\frac{\frac{1}{\mathbf{R}}M}{b\log_2 l}+\frac{L}{P}+X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right)\leq\frac{M}{b\log_2 l}+\frac{L}{P}$$

For the model using the band width

$$\frac{\frac{1}{\mathbf{R}}M}{2B\log_2 l}+\frac{L}{P}+X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right)\leq\frac{M}{2B\log_2 l}+\frac{L}{P}$$

For the model using the signal to noise ratio

$$\frac{\frac{1}{\mathbf{R}}M}{B\log_2(1+10^{\frac{S}{10}})}+\frac{L}{P}+X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right)\leq\frac{M}{B\log_2(1+10^{\frac{S}{10}})}+\frac{L}{P}$$

5.2 Using a Fixed-Length Page

In this model we assume that the protocol has a fixed length page and that the compression and decompression is done for a large chunk of data M, in this situation another parameter should be taken into consideration, which is the page size m and the expression for the transmission time should now include the number of compressed pages that are sent over the communication medium. Thus the above inequalities can now be expressed as :

$$\frac{\frac{1}{\mathbf{R}}M}{m}\left(\frac{m}{D}+\frac{L}{P}\right)+X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right)\leq\frac{M}{m}\left(\frac{m}{D}+\frac{L}{P}\right)$$

For the model using the baud rate

$$\frac{\frac{1}{\mathbf{R}}M}{m}\left(\frac{m}{b\log_2 l}+\frac{L}{P}\right)+X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right)\leq\frac{M}{m}\left(\frac{m}{b\log_2 l}+\frac{L}{P}\right)$$

For the model using the band width

$$\frac{\frac{1}{\mathbf{R}}M}{m}\left(\frac{m}{2B\log_2 l}+\frac{L}{P}\right)+X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right)\leq\frac{M}{m}\left(\frac{m}{2B\log_2 l}+\frac{L}{P}\right)$$

For the model using the signal to noise ratio

$$\frac{\frac{1}{\mathbf{R}}M}{m}\left(\frac{m}{B\log_2(1+10^{\frac{S}{10}})}+\frac{L}{P}\right)+X\left(\frac{M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}+\frac{\frac{1}{\mathbf{R}}M}{\frac{8w}{1.5+\mathbf{R}^{-1}}}\right)\leq\frac{M}{m}\left(\frac{m}{B\log_2(1+10^{\frac{S}{10}})}+\frac{L}{P}\right)$$

6 The Experiment

In our experiment towards the goal of establishing a reasonable tool for the design engineer, we offer the choice for either making a decision to use a compression/decompression scheme given a certain situation, i.e, a fixed set of physical layer parameters and a certain size of a chunk of data, or choosing to optimize { obtain the threshold of} a certain parameter, such that we can use compression for all values of the parameter that are less than this threshold, as it gives a less total cost than the

technique that does not use compression. The user is given the choice of choosing any one of the four different ways of modeling the cost function, to maximize the number of parameters that one can deal with. Thresholds are found by solving the above inequalities for the parameters that are to be optimized for minimizing the total communication cost.

The results of running the experiment are displayed both theoretically and realistically. In the theoretic solution, the input file to be transmitted is assumed to be a plain text, thus assuming no prior information about the kind of data that are transferred in the network, then, a more realistic { either a decision or a value } solution is given by calculating the actual compression and decompression runtimes by running the LZW algorithm on them and observing the times and the compression ratio.

Our algorithm is run on a variety of data types, in the first two examples the algorithm is run on image data. These image data are inherently different as the first one contains a lot of information and details, however, the second is mainly a few number of dots in a planar surface, it is not surprising then to know that the compression ratio in the second example turned to be equal to 48 !!, especially when we remember the adaptive characteristics of the LZW algorithm. The compression ratio in the first one was equal to 5. This fact contributed to the difference in the thresholds and decisions between the "theoretic" and the "realistic" approaches to finding the required limits and decisions. The following are snap shots of three different runs for the system, the first two are for the complex image, the third is for the simple one :

```
>> project.e
Enter various input values prompted for.

Decision or Optimization? [d/o]:o

Desired Model?
[1=using data rate, 2=baud rate, 3=bandwidth, 4=using sig-noise ratio]:3

Data Size? 8389192
Cycle Rate? 14286000
Network/CPU time cost ratio? 5
Theoretical Compression Ratio? 1.8
Observed Compression Ratio? 5.156
Observed Compression Time? 1.3
Observed Decompression Time? 1.2
Levels of Discretization? 2

Theoretical Results: If band width < 1588559.073359 Hz then compress.
Simulation Results:  If band width < 270484.731978 Hz then compress.

>>
>> project.e
Enter various input values prompted for.

Decision or Optimization? [d/o]:d

Desired Model?
[1=using data rate, 2=baud rate, 3=bandwidth, 4=using sig-noise ratio]:4
```

```
Data Size? 8389192
Cycle Rate? 14286000
Network/CPU time cost ratio? 5
Theoretical Compression Ratio? 5
Observed Compression Ratio? 5.156
Observed Compression Time? 1.3
Observed Decompression Time? 1.2
Signal to Noise Ratio in decibels? 5
Medium Bandwidth? 1000000

Theoretical Results: Compression would cost less.
Simulation Results: Compression would cost more.

>>
>> project.e
Enter various input values prompted for.

Decision or Optimization? [d/o]:o

Desired Model?
[1=using data rate, 2=baud rate, 3=bandwidth, 4=using sig-noise ratio]:1

Data Size? 1966640
Cycle Rate? 14286000
Network/CPU time cost ratio? 5
Theoretical Compression Ratio? 1.8
Observed Compression Ratio? 48.468
Observed Compression Time? 0.4
Observed Decompression Time? 0.2

Theoretical Results: If data rate < 3177118.146718  bps then compress.
Simulation Results:  If data rate < 642021.316608  bps then compress.

>>
```

We then proceed in the experiment to try different kind of data, we try executable files and observe the results of running our toolbox. The following is a sample run for the system on a file of executable commands.

```
>> project.e
Enter various input values prompted for.

Decision or Optimization? [d/o]:d

Desired Model?
[1=using data rate, 2=baud rate, 3=bandwidth, 4=using sig-noise ratio]:4

Data Size? 745472
Cycle Rate? 14286000
```

```
Network/CPU time cost ratio? 5
Theoretical Compression Ratio? 1.8
Observed Compression Ratio? 1.526
Observed Compression Time? 0.5
Observed Decompression Time? 0.3
Signal to Noise Ratio in decibels? 30
Medium Bandwidth? 3000

Theoretical Results: Compression would cost more.
Simulation Results: Compression would cost more.

>>
```

It is noticed that for a "nice" collection of information, which includes a lot of repetitiveness, the compression ratio is maximum, while it decreases for other types. The fact that there is sometimes a discrepancy between the realistic and the theoretic values is because the theoretic approach assumes a "perfect" media when it calculates the runtime for compression, however, this is not the case when performing the actual compression in software on a down-to-earth Vax workstation. Also the wide variations in the compression ratios should be taken into consideration.

7 Conclusions

We discussed the issue of efficient and adaptive transmission mechanisms over possible physical links between the operator and the system. A tool was developed for making decisions regarding the flow of control sequences and data from and to the operator. The decision of compressing data and commands is discussed in details, a yes/no decision box and an optimizing tool for finding the appropriate thresholds for a decision were implemented. Physical parameters are taken into consideration while developing our decision system. Also, the compression parameters relationships and different compression techniques suited for the task are developed, with an emphasis on adaptive ones that accommodate various data and control patterns. Theoretical analysis is performed to develop mathematical models for the optimization algorithm. Simulation models are also developed for testing both the optimization and the decision tool box. Our system is tested through a series of simulations and a comparison is performed against the theoretical results for some data and control sequences.

References

[1] V. Cappellini, *Data Compression and Error Control Techniques with Applications*, 1985.

[2] W.K. Pratt, *Image Transmission Techniques*, 1979.

[3] T.A. Welch, " A Technique for High Performance Data Compression", *IEEE Computer*, June 1984, pp. 8-19.

[4] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", *IEEE Trans. Information Theory*, Vol. IT-23, No.3, May 1977, pp. 337-343.

[5] J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding", *IEEE Trans. Information Theory*, Vol. IT-24, No.5, Sept. 1978, pp. 5306.